Github Actions Quiz -Answers

***Q1: In the context of*Github Actions*, what’s a workflow? How does it take form*on Github*and*on your repository*?***

* *Workflows: consist of jobs, which run concurrently by default.*
* *Each job should represent a separate part of your workflow.*
* *For example, you could have one job for running your tests, another for releasing your software, and a another one for deploying to your production environment.*
* *You can configure jobs to depend on the success of other jobs in the same workflow. For example, failing tests can prevent deploying to production.*

*-----------------------------------------------------------------------------------------------------------*

***Q2: Find and strikethrough the invalid key(s) for the Github workflow syntax in the list below:***

* runs-on: *(The runs-on attribute indicates the environment that the GitHub Action itself will run within. You can think of every action as being run within its own little container. It’s the type of machine to run the job on, but the action itself is executed in a container running on that platform)*
* container
* env: (*A map of environment variables that are available to the steps of all jobs in the workflow. You can also set environment variables that are only available to the steps of a single job or to a single step.)*
* with : *(provides further refined detail about the action that is going to be executed. For example, if you are executing a Jave Code, the ‘with’ specifies the Java version.)*
* ~~in~~ :

-------------------------------------------------------------------------------

***Q3: When talking about*CI/CD*, what’s an artifact? Why is it important for the*continuous deployment*part of a*CI/CD workflow*? :***

* *An artifact is a deployable component of your application.*
* *A release is a collection of artifacts in your DevOps CI/CD processes.*
* *In CI/CD systems, artifacts refer to the output of your build process that you would deploy.*
* *This could be a jar, tar or binary. You then deploy these artifacts onto your production servers.*
* *For pushes and pull requests, GitHub stores artifacts for 90 days.*
* *The retention period for a pull request restarts each time someone pushes a new commit to the pull request.*

----------------------------------------------------------------------------------------------------

**Q4: Why is a versioning tool like git so indispensable for CI/CD?**

*Three reasons why a Version Control tool is important for CI/CD*

1. *Avoiding dependency issues in containerized applications*
2. *Increases deployment performance*
3. *Supports building more reliable applications*

***Each reason explained here:***

1. ***Avoiding dependency issues in containerized applications:***

* *For DevOps, finding the balance between moving quickly and maintaining application reliability is crucial*
* *Part of that means cultivating traceability, gaining visibility into the changes made to the code and understanding how those changes affected application performance.*
* *Access to the different code iterations can reveal where new changes exposed dependencies that clash with other parts of the code.*

1. ***Increases deployment performance:***

* *the ability to more easily view and understand how changes in one part of the code caused problem across other parts of the code or even the whole application.*
* *It has also to do with how the VCS enables coding practices like Continuous Integration and, as a result, Continuous Delivery/Deployment.*

1. ***Supports building more reliable applications:***

* *To build reliable applications, you must get rid of errors. in DevOps, change is often the source of failures and therefore, there’s a big focus on mitigating the risk of change.*
* *Important first step is usually making smaller changes more frequently, (hence the popularity of CI).*
* *Just as important, and even more so in an accelerated workflow, is tracking those changes so that everyone in the team is looking at the same thing and so that troubleshooting is easier in the case of failures.*

*--------------------------------------------------------------------------------------------*

**Q5: In the context of Github Actions, what’s a runner? What type of runner can you find/use?**

* *The runner is the application that runs a job from a GitHub Actions workflow.*
* *It is used by GitHub Actions in the hosted virtual environments, or you can self-host the runner in your own environment.*
* *There are three types runners: Ubuntuu, Windows and Mac Os runners.*

*----------------------------------------------------------------------------------------------------*

**Q6: Can you change the configuration of an hosted runner? What do you need to do if none of the hosted runner meet your needs?**

* *If the GitHub hosted runners do not fit your purpose, you can add self-hosted runners to a single repository.*
* *To add a self-hosted runner to a user repository, you must be the repository owner.*
* *For an organization repository, you must be an organization owner or have admin access to the repository.*
* *You can add a self-hosted runner to a repository, an organization, or an enterprise.*
* *If you are an organization or enterprise administrator, you might want to add your self-hosted runners at the organization or enterprise level.*
* *This makes the runner available to multiple repositories in your organization or enterprise, and also lets you to manage your runners in one place.*

*---------------------------------------------------------------------------------------------*

**Q7: Describe (in your own words) what the workflow below does with as much details as possible.**

name: Mark stale issues and pull requests:

**name:** (*refers to the name of the workflow as it will appear in the Actions tab of the GitHub repository.)*

**on:** *specifies the event that automatically triggers the workflow file.   
Such an event can, for example be a push event, so that the jobs run every time someone pushes a change to the repository. You can set up the workflow to only run on certain branches, paths, or tags.*

**schedule:** *(refers to how often and when the specific trigger event is fired. You can schedule a workflow to run at specific UTC times using POSIX cron syntax. Scheduled workflows run on the latest commit on the default or base branch. The shortest interval you can run scheduled workflows is once every 5 minutes.*

- cron: "30 1 \* \* \*" this refers to the minute (30) hour (1) and \*\*\* for the day

jobs:

* *composed of a collection steps, which in turn consisting of specific actions.*
* *Those actions can be shell scripts or other stand alone commands*
* *Jobs are triggered by an event and*

**stale:**

* *is a branch that has not had any commits in the previous 3 months.*
* *This generally indicates an old/unmaintained/not current branch. ...*
* *So a "stale git branch" is generally a branch of the repository that hasn't been touched in a long time*

**runs-on: ubuntu-latest**

* *(Configures the job to run and be executed on the latest fresh virtual machine (container)*
* *based on Ubuntu Linux OS based runner hosted by GitHub.)*

**Steps:**

* *(specify or define the order in which actions and commands are executed in order to fulfil a job;*
* *they contain a collection of actions, that enable you accomplish the specified job.)*

- uses: actions/stale@v1 *=> refers to the version of the stale*

with: *=> specifies the attributes of token, message and label this step is going to use, as follows*

repo-token: ${{ secrets.GITHUB\_TOKEN }}

stale-issue-message: 'Stale issue message'

stale-pr-message: 'Stale pull request message'

stale-issue-label: 'no-issue-activity'

stale-pr-label: 'no-pr-activity'